Learning Relational Affordance Models for Two-Arm Robots

Bogdan Moldovan[1] and Luc De Raedt[1]

***Abstract*— Affordances are used in robotics to model action opportunities of a robotic manipulator on an object in the environment. Previous work has shown how statistical relational learning can be used in a discrete setting to extend affordances to model relations and interactions between multiple objects being manipulated by a robotic arm and deal with environment uncertainty. In this paper, we first extend this concept of relational affordances to a continuous setting and then to a two-arm robot. A relational affordance model can first be learnt for one arm through a behavioural babbling stage, and then with the use of statistical relational learning, after constructing a symmetrical model for the other arm, two-arm manipulation actions can be modelled, where the arms can act sequentially or simultaneously. The model is evaluated in a two-arm action recognition task in a shelf object manipulation setting.**

I. Introduction

Recent advances in robotics have led to an increase in the number and capabilities of humanoid robots, *e.g.*, PR2, iCub, NAO, and other robotic platforms, which need to be able to reason, learn and manipulate their environment. This includes dealing with higher-level knowledge needed for planning and reasoning, for which logical approaches are effective, but also dealing with various kinds of uncertainty arising from noisy sensors or physical actuators for manipulation. For both these needs, *statistical relational learning* (SRL) [1], [2], [3], or more generally *probabilistic programming languages* (PPLs) [4], can be used. They combine logical representations, probabilistic reasoning and machine learning.

Humanoid robots need to be able to manipulate objects in their environment, *e.g.*, in a household environment such as a kitchen, living room, etc., and one their main characteristic is that they have two symmetrical arms for manipulation. Furthermore, background knowledge about the environment and objects to be manipulated should be considered (*e.g.*, humans are more likely to manipulate two interacting objects, or parts of the same object, when using both hands).

A promising approach for the development of humanoid robots' skills is learning *object affordances*. They capture *action opportunities* to structure the environment (i.e., what can be done with an object?) through the robot's available sensing and motor capabilities. They model relations between three variables: objects properties, actions, and effects [5], [6]. They are in accordance with the robotics developmental framework, which proposes acquiring new skills on top of old ones by experimentation in the environment [7].

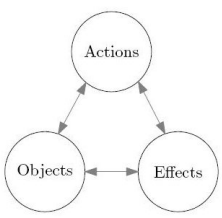
Recent work [8] extended affordances to relational affordances in a discrete setting by modelling interactions and spatial relations between multiple objects in the environment with the help of SRL. These models were used to generalise and perform inference [8]. In this paper, we first extend the concept of relational affordances to a *continuous* setting where object properties and effects can be modelled by continuous distribution random variables. The robot learns this one-arm model by exploring two-object interactions. By using SRL a symmetrical model is defined for the other arm, and then for actions requiring the use of both arms. We add constraints coming from the actions and the environment, building a model for two-arm relational affordances in a multiple object environment, for settings where the two-arm action effects are a combination of the single-arm ones.

We start with background in Section II, and describe the setting and approach in Section III. The extension of relational affordance models to the continuous domain, (i.e., Steps 1a and 1b in our pipeline) is shown in Section IV. Section V describes Step 2, our relational affordance model of two-arm actions. Section VI presents experimental results, obtained by Step 3. We conclude in Section VII.

II. Background

A. Affordance-based Models

Affordance models, based on a concept introduced by J. J. Gibson [9], are used to model the robot-world interaction by structuring the environment by capturing action opportunities. They define the relationships between the robot and its environment through the robot's available sensing and motor capabilities [6]. For this purpose, affordances model the correlations between the set of *objects (properties)* as being detected by the robot sensors: $O = \{o_1, ..., o_n\}$, the repertoire of *actions* available to the robot, $A = \{a_1, ..., a_n\}$, and the set of *effects* of performing those actions $E = \{e_1, ..., e_n\}$ as detected by the sensors as changes in object features. Figure 1 depicts a generic affordance model.



Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

Fig. 1. Affordances: relations between objects, actions, effects [5], [6].

[1]Bogdan Moldovan and Luc De Raedt are with the Department of Computer Science, Katholieke Universiteit Leuven, Belgium

Bogdan Moldovan is supported by IWT (agentschap voor Innovatie door Wetenschap en Technologie).

An affordance model is normally used for inferring one of the variables O, A, and E, when the other two are known. They can be used for imitation learning [5] or action prediction: computing the *maximum a posteriori probability* (MAP) estimate $\arg\max_A P(A|O,E) = \arg\max_A \frac{P(A,O,E)}{P(O,E)}$, given the values of O and observing the E.

B. Related Work

Related work includes research on tool use for robots, such as [10] which learns the tool affordances of an object from a human demonstration together with a set of robot experimentations based on an inductive logic programming algorithm. Research on behaviour-grounded tool affordances such as [11], [12] provides algorithms for the robot to learn the effects of its actions with given tools on other objects.

However, all these involve robots with only one end-effector. There are very few studies on two-arm robots. Among this, there is research on learning, representing and generalising a task which presents a programming-by-demonstration framework for extracting and generalising knowledge about a given task [13], and similarly a programming-by-demonstration framework for dual-arm manipulation tasks [14]. There is also research on motion planning for dual-arm manipulation and re-grasping tasks [15]. However, none of these use the concept of affordances, model or generalise over relations and interactions between manipulated objects and other objects in the environment, or build a two-arm manipulation model from a one arm affordance model obtained by environment experimentation and the use of background knowledge.

III. Manipulation Setting and Approach

A. Problem Statement and Approach

We tackle a table-top object manipulation scenario with multiple objects that can interact with one another during robot manipulation, *e.g.*, in Figure 2. The robot's task in this setting is to place the green bar, which it cannot reach directly, at the target location. It can achieve this by pushing simultaneously the two red objects with its two arms.

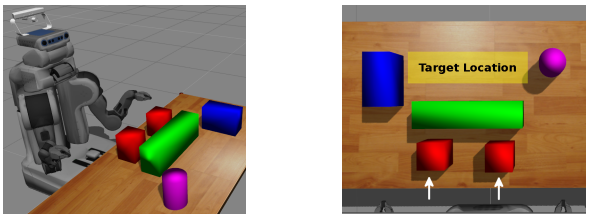


Fig. 2. Table-top scenario with two-arm actions for object placement.

Note that the use of both arms, whose actions can be either simultaneous or sequential, enables the robot to perform tasks not possible by the use of only one (*e.g.*, just pushing one of the red objects would not get the bar to the goal). By learning and using a two-arm relational affordance model, the robot can perform these even though it had never explored these settings during its one-arm behavioural babbling phase.

To tackle such object manipulation scenario, we need to solve three tasks (two learning tasks and an inference task):

Task (i) is learning a one-arm continuous relational affordance model: *given:* a) a set of corresponding O, A, E values collected from exploratory one-arm action executions in two-object environments, *find:* b) a continuous setting relational affordance model of one-arm actions

Task (ii) is learning a two-arm model: *given:* a) the one-arm affordance model learnt in Task (i), and b) a set of background rules constraining two-arm actions (*e.g.*, arms act on the same or on interacting objects), *find:* c) a relational affordance model of two-arm actions. Note that the robot can learn new affordances for objects for its two arms that were not possible with single arm actions.

Task (iii) is the inference task used to evaluate our model: *given:* a) the two-arm affordance model learnt in Task (ii), b) an initial scene from which the set of object properties values O can be extracted, and c) a target goal, given as E, *find:* d) the best action to execute to reach the goal, given by: $\arg\max_A P(A|O,E)$.

To solve these tasks, we propose a pipeline, as shown in Figure 3, with four steps: 1a) learn a Linear Continuous Gaussian (LCG) Bayesian Network (BN) from the exploratory data, 1b) from the LCG model, build a one-arm continuous domain relational affordance model in a PPL, 2) generalise the one-arm model to a two-arm model, and 3) perform action prediction using the two-arm model. Steps 1a and 1b together solve Task (i). The other two steps solve Tasks (ii) and (iii) respectively.

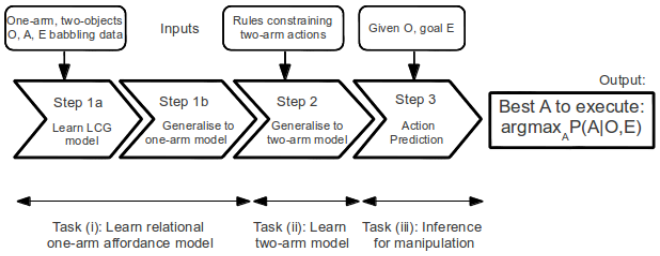


Fig. 3. Pipeline for table-top two-arm object manipulation.

B. Scenario Setting and Robot Skills

We now describe more precisely our setting, introduced in Figure 2. We employ the *PR2 humanoid robot* in the Gazebo simulator, in contrast with [8] which used the iCub simulator. We use both arms of the PR2 robot with the *arm_navigation* stack. Arm actions are performed by sending the arm to a goal location in Cartesian space, using inverse kinematics in order to plan a trajectory for the arm to reach a position as close as possible to the goal within a *2cm* tolerance.

We now describe the variables of our affordance model, illustrated in Figure 4 with the objects' position before (l) and after (r) an action (tap) execution.

The set of object properties O consists of the following: $shape$, and (the relational properties) relative distances along the x-axis ($distX$) and y-axis ($distY$) between the objects (in cm). As opposed to [8] we use a Cartesian coordinate system instead of a polar one, which facilitates modelling in a continuous domain setting (using Gaussian distributions for

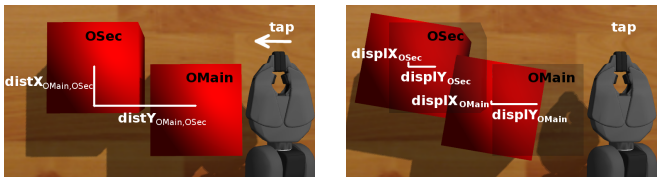


Fig. 4. Relational O before (l), and E after the action execution (r).

distances and displacements). We use four object shapes, the two used in [8], denoted *cube* and *prism*, and two additional ones: a *cylinder*, to increase object interaction variation, and a long *bar*, for two-arm manipulation. All object shapes are shown in Figure 2. $distX$ and $distY$ are measured centre-to-centre as in Figure 4(l). The centre position is obtained by querying the Gazebo simulator. The x and y-axes correspond with the Gazebo x and y-axes from the robot's viewpoint.

A can be one of two basic single arm core motor actions: `tap` (right-to-left hand movement for the right arm, left-to-right for the left) and `push` (away movement for both arms). Actions move the arm over a preprogrammed distance and orientation, but could be parameterised without posing problems to our approach. For all the shapes except *bar*, the action is executed in the middle of the object. For *bar*, each arm acts towards its end of the object.

E corresponds to differences in object attributes before and after the action is performed. We use the displacements along the x-axis ($displX$) and y-axis ($displY$) of the centre of each object. These are shown in Figure 4(r), which overlays the initial objects' positions over their final positions.

As in our setting the robot is not mobile and each arm has a specific action range, each $a_i \in A$ can be performed when an object is located in a specific action space. An object can be acted upon by both arms, by one arm but not the other, or it can be completely out of the reach of the robot. The action space is learnt during the babbling phase. If the exploratory arm action on an object fails because no inverse kinematics solution was found, then that object is not in that arm's action space. We will show later how any spatial constraints, such as action space, can be modelled with logical rules.

C. Contributions

The main contributions of this paper are: i) extending the model of [8] to a continuous domain setting, and ii) using this and SRL methods to create a relational affordance model for two-arm actions, for settings where these can be approximated by a combination of the two single-arm actions composing them. The arms may act simultaneously or sequentially, and the robot is given background knowledge about possible actions in its environment. Few studies have investigated two-arm manipulation, and our contribution is to use SRL to generalise and build a higher-level model for a set of two-arm actions settings in a household environment.

IV. Learning Relational Affordances in a Continuous Setting

We will describe our extension of the relational affordances in the multiple-object table-top manipulation setting of [8] in order to support models with continuous distribution random variables. To learn the single-arm relational affordance model, data (corresponding O, A, E values) are obtained from a behavioural babbling stage with the right-arm only. Pairs of objects are placed in front of the robot at various positions. The robot executes one of its actions A on one object (named: main object, O_{Main}). O_{Main} may interact with the other object (secondary object, O_{Sec}) causing it to also move. Figure 4 shows such a setting. The robot executed 300 such exploratory actions, obtaining 300 sets of O, A, E values. One such set, for the action shown in Figure 4, is shown in Table I.

TABLE I

Collected O, A, E data for the tap action in Figure 4

Object Properties	Action	Effects
$shape_{O_{Main}}$: $cube$ $shape_{O_{Sec}}$: $cube$ $distX_{O_{Main},O_{Sec}}$: $7.00cm$ $distY_{O_{Main},O_{Sec}}$: $17.00cm$	tap	$displX_{O_{Main}}$: $0.40cm$ $displY_{O_{Main}}$: $7.23cm$ $displX_{O_{Sec}}$: $0.39cm$ $displY_{O_{Sec}}$: $4.38cm$

A. Learning a Linear Conditional Gaussian BN

We now present Step 1a of our pipeline, learning a *Linear Continuous Gaussian (LCG) Bayesian Network* (BN) [16] from the exploratory data. The LCG BN specifies a distribution over a mixture of discrete and continuous variables. We allow the modelling of $distX$, $distY$, $displX$ and $displY$ with continuous distribution random variables. In an LCG, a discrete random variable may have only discrete parents, while a continuous random variable may have both discrete and continuous parents. A continuous random variable (X) will have a single Gaussian distribution function whose mean depends linearly on the state of its continuous parent variables (Y) for each configuration of its discrete parent variables (U) [16]. This LCG distribution can be represented as: $P(X|Y = y, U = u) = \mathcal{N}(M(u) + W(u)^T y, \sigma^2(u))$, with M a table of mean values, W a table of regression (weight) coefficient vectors, and σ a table of variances (independent of Y). [16]

Our setting can be modelled by an LCG BN with $displX$, $displY$, $distX$ and $distY$ approximated by conditional Gaussian distributions over the short distances over which objects interact. We show later how to enforce these distances by adding logical rules. The experiments will show that this approximation is better than the discretisation in [8].

The LCG model of our setting is shown in Figure 5, where discrete random variables are represented by a single ellipse, and continuous ones by a double ellipse. $displX_{O_{Main}}$ and $displY_{O_{Main}}$ only depend on A and the object $shape$ since the hand is moved over a preprogrammed distance (with a given tolerance). $displX_{O_{Sec}}$ and $displY_{O_{Sec}}$ depend on both the relative distance O_{Sec} is away from O_{Main} and the shapes of both objects.

The LCG parameters are learnt from the collected babbling data (*e.g.*, Table I) by using the maximum likelihood parameter estimation from the BNT toolbox [17] for Matlab. *E.g.*, during our *tap* action for two interacting cubes (as in

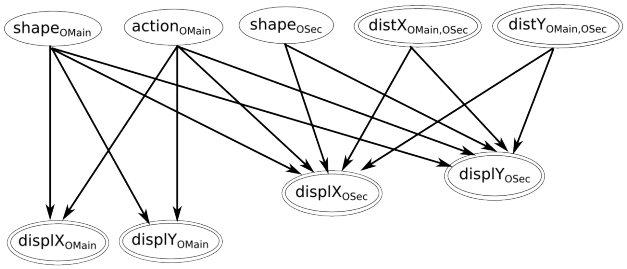


Fig. 5. LCG BN model for two-object interaction

Figure 4), the displacement of O_{Sec} on the y-axis is (in cm):

$$\mathcal{N}(19.92 - 0.05 * distX_{O_{Main},O_{Sec}} - \\ -0.86 * distY_{O_{Main},O_{Sec}}, 0.17). \quad (1)$$

Intuitively this makes sense: the second cube is moved along, so we expect the learnt coefficient of $distY$ to be close to -1, but also to depend a little bit on $distX$ if the objects are not aligned, as in Figure 4. Also intuitively, the mean coefficient generally depends on the widths of the shapes ($15cm$ for cubes), as well as the preprogrammed action distance.

B. Probabilistic Programming Languages

A PPL is a programming language specifically designed to efficiently describe and reason with probabilistic relational models. A PPL program consists of a set of probabilistic facts and a set of logical rules expressing *domain knowledge* and *constraints*. In [8] relational affordances were modelled using the PPL ProbLog [4]. Here, since we deal with continuous distribution random variables, modelled by normal distributions, we use our new state-of-the-art PPL Distributional Clauses (DCs) [18], a continuous extension of ProbLog.

We first review basic concepts of logic programming: An atom $pred(t_1, ..., t_n)$ consists of a predicate $pred/n$ of arity n and t_i terms. A term is either a (lowercase) *constant*, (uppercase) *variable*, or functor $func/n$ applied on n terms. A *ground* atom does not contain any free variables. A *definite clause* is an expression of the form $h \leftarrow b_1, ..., b_n$, where h and b_i are atoms, h is the head of the clause and the b_i the body. It states that h is true whenever all b_i are true. If $n = 0$ we have a fact $f \leftarrow$, i.e., f is true. A *substitution* $\theta = \{X_1 = t_1, ..., X_n = t_n\}$ maps each variable X_i to a term t_i. Applying a substitution θ to an atom a yields $a\theta$, where each occurrence of X_i in a is replaced with t_i.

DCs [18] are an extension of the distribution semantics of [19]. DCs allow one to define random variables with any distribution, discrete or continuous. A DC is an expression of the form $h \sim \mathcal{D} \leftarrow b_1, ..., b_n$, where b_i are atoms and $\sim$ a binary predicate written in infix notation. In a DC, each ground instance of the clause $(h \sim \mathcal{D} \leftarrow b_1, ..., b_n)\theta$, for a substitution θ, defines the random variable $h\theta$ being distributed according to distribution $\mathcal{D}\theta$ when all $b_i\theta$ hold.

Now we can proceed to Step 1b of our pipeline, modelling the LCG using DCs to generalise to a one-arm relational affordance model. We generalise over the number of objects as in [8], by introducing variables for objects (*e.g.*, $displX(O_{Main})$ for the $displX_{O_{Main}}$ in the LCG), and so build a general multiple object PPL model from the two-object LCG BN. We illustrate the modelling with examples.

To model the shape of an object being randomly chosen from our set of 4 shapes, each with 25% probability:

$$\texttt{shape(O)} \sim \texttt{finite}([\tfrac{1}{4} : \texttt{cube}, \tfrac{1}{4} : \texttt{prism}, \tfrac{1}{4} : \texttt{bar}, \tfrac{1}{4} : \texttt{cyl}]) \\ \leftarrow \texttt{obj(O)}.$$

variable O universally quantified over the set of all objects.

The term $\mathcal{D}$ that represents the distribution can be non-ground: values, probabilities, or distribution parameters can be related to conditions in the body. Additionally, the term $\simeq(d)$ represents the value that the random variable d takes.

So to transform the LCG Equation 1 in DCs, one writes:

$$\texttt{displY}(\texttt{O}_{\texttt{Sec}}) \sim \texttt{gaussian}(\texttt{Mu}, 0.17) \leftarrow \texttt{act}(\texttt{tap}, \texttt{O}_{\texttt{Main}}), \\ \simeq(\texttt{shape}(\texttt{O}_{\texttt{Main}})) = \texttt{cube}, \simeq(\texttt{shape}(\texttt{O}_{\texttt{Sec}})) = \texttt{cube}, \\ \simeq(\texttt{distX}(\texttt{O}_{\texttt{Main}}, \texttt{O}_{\texttt{Sec}})) = \texttt{DX}, \simeq(\texttt{distY}(\texttt{O}_{\texttt{Main}}, \texttt{O}_{\texttt{Sec}})) = \texttt{DY}, \\ \texttt{Mu is } 19.92 - 0.05 * \texttt{DX} - 0.86 * \texttt{DY}.$$

meaning for a tap action, if the two shapes are cubes, $displY$ of O_{Sec} is distributed according to a Gaussian with mean given by Mu, as in Equation 1.

We can use definite clauses to model that the above Gaussian approximation holds only over small distances ($10cm$ on the motion axis and while there is overlap on the orthogonal axis), while over big distances there will be no effect on O_{Sec}. For our two cubes running example:

$$\texttt{approx\_ok(tap, cube, cube, DX, DY)} \leftarrow \texttt{DY} > 15, \\ \texttt{DY} < 25, \texttt{DX} > -15, \texttt{DX} < 15.$$

where $15cm$ is the smallest centre-to-centre distance between two cubes. We then just need to add $\texttt{approx\_ok(tap, cube, cube, DX, DY)}$ to the body of the DC clause defining displY above. Similar rules can be added to enforce the action space.

At this point we have all the tools to fully model the right arm relational affordance model with the parameters learnt as in Section IV-A. Once the program is defined, the inference algorithm based on sampling from [18] or [20] is used to compute the probability of a user's query. For example, assuming two cubes o_1 and o_2, one can ask for the probability of the y-axis displacement of o_2 being greater than $3cm$ given some distances between o_1 and o_2:
$P(\simeq(displY(o_2)) > 3 | act(tap, o_1), \simeq(distX(o_1, o_2)) = 4, \simeq(distY(o_1, o_2,)) = 2)$.

V. Learning Affordance Models for Two-Arm Robots

We now have a relational affordance model for the right arm, so we can proceed to Step 2, creating the two-arm model. For this, we need to create first the model for the left arm. Given the symmetry of the robot, the model for the left arm is equivalent to the model for the right arm mirrored through the plane perpendicular to the table that passes through the centre of the robot. For our model (and Gazebo framework), in the left-arm model all the x-axis values are the same as for the right-arm model, but the y-axis values are the negative of their right-arm equivalent.

In our DC framework, the $displX$ and $displY$ random variables for left and right arm actions need to be defined by different probability distributions. So, we need to add an

extra term to our $displX$ and $displY$ atoms to signify the arm performing the action. At this point, we can automatically generate the PPL code for the left arm. For our running example, the equivalent code for Equation 1 for the left arm:

$\texttt{displY}(\texttt{O}_{\texttt{Sec}}, \texttt{left}) \sim \texttt{gaussian}(\texttt{Mu}, 0.17) \leftarrow$
$\texttt{act}(\texttt{tap}, \texttt{left}, \texttt{O}_{\texttt{Main}}), ...$
$\texttt{Mu is } -19.92 + 0.05 * \texttt{DX} + 0.86 * \texttt{DY}.$

Now we are ready to model two-arm actions, which we will define with the atom $\texttt{twoArmA}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R})$, where A_L and A_R signify the left and right arm actions, and O_L and O_R the objects the arms act on:

$\texttt{twoArmA}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R}) \leftarrow \texttt{act}(\texttt{A}_\texttt{L}, \texttt{left}, \texttt{O}_\texttt{L}),$
$\texttt{act}(\texttt{A}_\texttt{R}, \texttt{right}, \texttt{O}_\texttt{R}).$

A. Adding Task Constraints in Environment

In a normal household environment there are many objects present, so it becomes infeasible for a robot to infer the success probability of acting on each of them when given a task. In our relational affordance model, we want to narrow down the state space search of the robot. So we can add constraints to our two-arm SRL model, which can be done by adding logical rules.

We first want to define some spatial constraints by defining the $\texttt{spCnstr}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R})$ atom, to be added to the body defining $twoArmA$ above. If the hard rules of the constraints are not met, the probability of that two-arm action will be zero. For example, the left arm should not act on an object more to the right than the one acted on by the right arm (i.e., y-distance between O_L and O_R positive):

$\texttt{spCnstr}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R}) \leftarrow \simeq(\texttt{distY}(\texttt{O}_\texttt{L}, \texttt{O}_\texttt{R})) > 0.$

More rules can be added to the body of the $spCnstr$ clause (*e.g.*, one object should not be right behind the other).

Generally speaking, we want our relational affordance model to allow modelling of household environments, where objects are intended for human use and the robot tasks are similar to human tasks. We want to show how our model can be augmented for this type of environments by adding logical rules. In these environments, usually two-arm human actions fall into one of the following three categories:

1) Both arms act on the same object
2) One arm acts on an object, while the other acts on an object that interacts with the first object
3) The two arms act on two different objects that both interact with the same third object

The first type of two-arm action corresponds for example to unscrewing a bottle cap, or carrying around a tray. The second one corresponds with tool use by one arm, similar with [10], [11] while the other arm supports or manipulates the object, *e.g.*, hammering a nail. Lastly, the third type corresponds with both arms using tools to act on the same object, *e.g.*, using a fork and knife on food.

We can do this in a more abstract manner by defining a two-arm constraints $taCnstr$ atom to be added to the body of $twoArmA$. In this case, $taCnstr$ will be defined by several clauses, at least one of them needing to be true for $twoArmA$ to have a non-zero probability. The first two types of two-arm actions can be modelled as:

$\texttt{taCnstr}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R}) \leftarrow \texttt{O}_\texttt{L} == \texttt{O}_\texttt{R}.$
$\texttt{taCnstr}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R}) \leftarrow \texttt{O}_\texttt{L} \neq \texttt{O}_\texttt{R}, \texttt{canInt}(\texttt{A}_\texttt{L}, \texttt{O}_\texttt{L}, \texttt{A}_\texttt{R}, \texttt{O}_\texttt{R}).$

where $canInt$ holds if the objects are close enough to allow interaction given the two actions. The third type is modelled similarly as the second one, where $canInt$ now also considers a third object.

The use of these rules will restrict the search space of the robot when doing inference. Note that depending on the environment, different other background rules can be added as well (*e.g.*, for object search, many objects are manipulated and moved away with both hands).

B. Sequential or Simultaneous Use of Arms

We finally need to model the effects of both arms acting on the environment. Note that the left and right arm actions can take place simultaneously (*e.g.*, carrying a tray), or sequentially (*e.g.*, one arm picks up an object and passes it to the other arm for manipulation). For this, we need again to add an extra term T (with values 1 or 2) to our atoms signifying the time-step. Furthermore, we model the overall displacement (dX and dY) of an object during a time-step as the sum of the displacements caused by left and right arm actions during that time-step. For example, along the y-axis:

$\texttt{dY}(\texttt{Obj}, \texttt{D}, \texttt{T}) \leftarrow \texttt{D is } \simeq(\texttt{displY}(\texttt{Obj}, \texttt{left}, \texttt{T})) +$
$\simeq(\texttt{displY}(\texttt{Obj}, \texttt{right}, \texttt{T})).$

Note that for one or both arms, the displY distributions might not be defined (if the object is not manipulated or does not interact with one that is). So we need to also add the 3 definite clauses for dY corresponding to these cases.

At this point, we can already do action prediction for two-arm simultaneous actions A_L and A_R, since this lasts only one time-step. For this, given our overall DCs model for the two arms, we just need to calculate the MAP estimate: $\arg\max_{A_L, O_L, A_R, O_R} P(twoArmA(A_L, O_L, A_R, O_R) | O, E)$ by doing inference in our PPL program.

For sequential actions, we have a two-step planning problem, which needs to be modelled in our probabilistic setting. There are several ways of achieving this, we chose to model it in our PPL with a probabilistic STRIPS formalism as in [21], [22], where in comparison to classical STRIPS, each action has several outcomes, each associated with a probability that it might occur.

For illustration, we expand on our running example with two cubes from Figure 4 by considering sequential two-arm actions. We will refer to the right cube as o_r and the left as o_l. The first action will be the right-hand tap on o_r as in Figure 4. We assume the second action will be a left-hand push on o_l for $7cm$, in which case the two objects will not interact during the second action.

The first task consists of defining the states. In our particular table-top setting, we can define the state the objects are in by their defined (affordance) object properties. So, a state will consist of a conjunction of grounded shape, distX and distY atoms. In our example from Figure 4, with collected data O, A, E as in Table I, the initial state S_0 in STRIPS notation is: $shape(o_r, cube)$,$shape(o_l, cube)$,$distX(o_r, o_l, 7)$, $distY(o_r, o_l, 17)$. The left column of Table II shows S_0.

TABLE II

EXAMPLE STATES FOR TWO-ARM ACTIONS.

S_0	S_1	S_2
$shape(o_r, cube)$	$shape(o_r, cube)$	$shape(o_r, cube)$
$shape(o_l, cube)$	$shape(o_l, cube)$	$shape(o_l, cube)$
$distX(o_r, o_l, 7)$	$distX(o_r, o_l, 6.99)$	$distX(o_r, o_l, 13.99)$
$distY(o_r, o_l, 17)$	$distY(o_r, o_l, 14.15)$	$distY(o_r, o_l, 14.15)$

To finish modelling the states, we are left with defining the relation between our `displX` and `displY` object displacements (affordance) effects and our state literals. For this we observe that the x-axis and y-axis distances between objects in the next state can be defined in terms of the ones in the previous state and the object displacements. For example:

$\texttt{distY}(\texttt{O}_1, \texttt{O}_2, \texttt{D}, \texttt{T}) \leftarrow \texttt{PrevT is } T-1,$
$\texttt{distY}(\texttt{O}_1, \texttt{O}_2, \texttt{PrevD}, \texttt{PrevT}), \texttt{dY}(\texttt{O}_1, \texttt{Y}_1, \texttt{T}),$
$\texttt{dY}(\texttt{O}_2, \texttt{Y}_2, \texttt{T}), D \texttt{ is PrevD} + \texttt{Y}_2 - \texttt{Y}_1.$

so the overall state model can only be represented by using the shapes and relative distances atoms.

For example, to compute $distY$ at S_1 between o_r and o_l: $17 + 4.38 - 7.23 = 14.15$. This can be seen in the second column of Table II. The rest of S_1, and the final goal state S_2 after the two actions can be computed similarly.

Using the clause above and our DCs model derived from the LCG, we have a state-transition model. Given a state as a conjunction of grounded `shape`, `distX` and `distY` atoms, and an action, we can compute the next state `distX` and `distY` (by first computing `displX` and `displY`) and their probability distributions.

This model also gives an action representation. An action representation consists of a set of rules, each rule a four-tuple: $(action, precond, effects, prob)$. The precondition is a conjunction of the `shape`, `distX` and `distY` atoms for the objects whose relative distances change during the action. The effect is a delete list containing a conjunction of the `distX` and `distY` in the precondition, and an add list containing a conjunction of `distX` and `distY` with the new distance values given by the model (*e.g.*, as in Equation 1). The probability is given by the distribution of the DC clauses.

For our running example, the (grounded) action representation of the tap action from S_0 to S_1 is:

action: $tap(o_r)$
precond: $shape(o_r, cube), shape(o_l, cube),$
$distX(o_r, o_l, 7), distY(o_r, o_l, 17)$
effects: $\neg distX(o_r, o_l, 7), \neg distY(o_r, o_l, 17),$
$distX(o_r, o_l, 6.99), distY(o_r, o_l, 14.15)$

At this point we can use our DCs program and state-transition model. Given S_0 and S_2, we can find the best set of left and right arm actions by forward or backward state-space search over the two time steps, and also in our model we can compute $P(twoArmA(A_L, O_L, A_R, O_R)|S_0, S_2)$ for any action A_L and A_R and objects O_L and O_R.

Note that to restrict the size of each state, and that of the action representation, in this two-step planning problem we can restrict the states to contain only the subset of objects close enough to be manipulated or to interact with objects that are manipulated.

We now have a full two-arm relational affordance model for settings where the two-arm manipulation can be approximated by a combination of the one-arm actions composing it, which we can evaluate in an action prediction setting, which will be Step 3 of our pipeline.

VI. EVALUATION AND RESULTS

We want to investigate whether our two-arms probabilistic relational affordance model can be used successfully in a table-top object manipulation setting. We do this in the context of action prediction, where the robot needs to pick the object(s) to act on and the best left and right arm actions to achieve the required effects. We want to find out: (i) Does our continuous domain model have a higher action prediction rate than the previous discretised model, (ii) Can the robot pick the correct object(s)? (iii) Can it pick the correct left and right arm actions?, and (iv) Can the robot handle tasks suitable for two-arm manipulation?

We use a table-top setting based on the multiple object action prediction setting of [8]. Objects are placed on the table in front of the robot in two layers as in [8]. Each layer has either one *bar*, or three objects that can be of any of the other three shapes. We generate random shapes for the objects. We ignore the trivial setting with two bars (we want more objects than in babbling phase), so we generate settings with either four or six objects. Objects in the front layer are always in the field of action of the robot, but not necessarily in the action space of both arms. The objects placed behind might interact with them during an action. All objects are randomly placed within certain margins. Figure 6(l) shows one such placement. In each placement we extract O, which is a set containing the values of the shapes of the objects and of the $distX$ and $distY$ for all (ordered) pairs of objects.

We then execute all possible combinations of two-arm actions that meet our two-arm action constraints from Section V-A with the PR2 (but we ignore the trivial case where both arms tap the same object, where final configuration of objects would be similar to initial). We retain all those where the inverse kinematics succeeds and the arms act on the object(s). We get a dataset containing a set of real-world goal effects E matching the O. E contains a set of $displX$ and $displY$ values for all the objects in the scene. Figure 6(r) shows one such goal, which was obtained from the initial setting in Figure 6(l) by executing the two sequential actions: left-arm tap on the (left) blue prism (o_1), right-arm push of the red cube (o_2). We use our model to infer the most likely two-arm actions to achieve E, and compare these against the ground truth actions performed when obtaining the dataset.

We generated 100 such settings. One such setting, for Figure 6, where we preprocessed the E to transform the object displacements in a goal state consisting of relative object distances is:

Initial: $shape(o_1, prism), shape(o_2, cube), shape(o_3, pr...,$
$distX(o_1, o_2, -1.51), distX(o_1, o_3, -1.37), ...,$
$distY(o_1, o_2, -20.59), ...$
Goal: $distX(o_1, o_2, 5.68), distX(o_1, o_3, -2.21), ...,$
$distY(o_1, o_2, -15.60), ...$

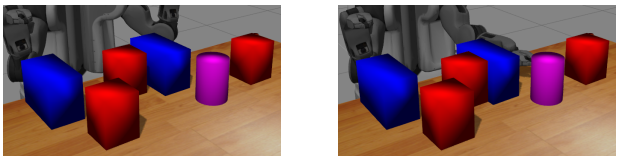


Fig. 6. Initial object placement (l), and its goal final object locations (r)

We used the PR2 robot in the Gazebo simulator as described in Section 1.3. Given that the inverse kinematics trajectory planning has a *2cm* tolerance around the desired goal, our target displacement effects E' in our continuous domain setting are the $4cm$ interval centered on the ground truth dataset values E. We use the SRL model to predict the left and right arm action-object pairs by calculating $\arg\max_{A_L,O_L,A_R,O_R} P(twoArmA(A_L,O_L,A_R,O_R)|O,E')$ and compare these to the ground truth action-object pairs. Table III summarises the results of the experiments.

TABLE III

TWO-ARM MODEL ACTION PREDICTION.

	Total exp.	Success	Pct.
Correct two-arm object(s)/actions	100	68	68%
Correct manipulated objects	100	74	74%
Correct left/right actions	100	68	68%
Random choice			2.78%
Random choice given constraints			9.52%

The robot picks the correct two-arm actions and object(s) to act on in 68% of the cases. For comparison, the original discretised single-arm single-action affordance model was 58% accurate [8], and it used a simpler setting with only two shapes. This shows the answer to question (i) of our evaluation: our continuous extension is better suited for modelling this setting than our previous discretised model.

Furthermore, we predict the correct object(s) to act on in 74% of the cases. The pair of left/right actions are predicted correctly 68% of the time. This shows that our two-arm model can be used by the robot to infer which object(s) and which left/right actions to use to reach a goal in a manipulation setting (questions (ii), (iii) of our evaluation).

We also included in Table III the random prediction baselines. The probability of randomly picking the correct two-arm object(s)/actions is only 2.78%. This increases to 9.52% if we restrict the actions according to the task constraints introduced in Section V-A.

Qualitatively, many of the random settings in the dataset are good showcases for two-arm manipulation (question (iv)). The four-object settings are either similar to the one in Figure 2 if the *bar* is in the back layer, or if it is in the front layer by pushing it with both arms it can interact with two or even all three of the back layer objects which would not be the case with single-arm actions. Settings such as Figure 6 illustrate another two-arm scenario: one arm taps an object which by interacting with a second object makes the latter closer for the other arm to act on.

Experiments were run on computers with Intel Core $i5-2500$ $3.3GHz$ processors, $6MB$ cache, and $8GB$ memory. We implemented our model with DCs and used 10000 samples for computing the query probabilities. Each query $P(twoArmA(A_L,O_L,A_R,O_R)|O,E')$, equivalent to $\frac{P(twoArmA(A_L,O_L,A_R,O_R),E'|O)}{P(E'|O)}$ for better inference performance, took about 30 seconds.

VII. CONCLUSION AND FUTURE WORK

We have presented an extension of the relational affordance model for the continuous domain and then introduced the two-arm relational affordance model. We showed that such a model can be used for two-arm manipulation in a multiple object scene, as shown in our experiments on action recognition. Future work will investigate the use of different additional spatial relations and using other background rules for improving action recognition performance, and more complex environments. We also want to investigate sequences of two-arm actions to achieve a task.

REFERENCES

[1] L. Getoor and B. Taskar, *Introduction to statistical relational learning*. The MIT press, 2007.

[2] L. De Raedt and K. Kersting, "Probabilistic inductive logic programming," in *Prob. Ind. Log. Progr.*, 2008, pp. 1–27.

[3] L. De Raedt, *Logical and Relational Learning*. Springer, 2008.

[4] L. De Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic Prolog and its application in link discovery," in *IJCAI*, 2007.

[5] M. Lopes, F. S. Melo, and L. Montesano, "Affordance-based imitation learning in robots," in *IROS*, 2007, pp. 1015–1021.

[6] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory-motor coordination to imitation," *IEEE Transactions on Robotics*, vol. 24, pp. 15–26, 2008.

[7] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: A survey," *Connection Science*, vol. 15, pp. 151–190, 2003.

[8] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *ICRA*, 2012.

[9] J. J. Gibson, *The Ecologial Approach to visual perception*. Boston: Houghton Mifflin, 1979.

[10] S. Brown and C. Sammut, "A relational approach to tool-use learning in robots," in *Inductive Logic Programming*, 2013.

[11] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *ICRA*. IEEE, 2005, pp. 3060–3065.

[12] J. Sinapov and A. Stoytchev, "Learning and generalization of behavior-grounded tool affordances," in *ICDL*, 2007, pp. 19–24.

[13] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 286–298, 2007.

[14] R. Zöllner, T. Asfour, and R. Dillmann, "Programming by demonstration: dual-arm manipulation tasks for humanoid robots," in *IROS*. IEEE, 2004, pp. 479–484.

[15] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *IROS*. IEEE, 2009, pp. 2464–2470.

[16] U. B. Kjærulff and A. L. Madsen, "Probabilistic networks-an introduction to bayesian networks and influence diagrams," *Aalborg University*, 2005.

[17] K. Murphy *et al.*, "The bayes net toolbox for matlab," *Computing science and statistics*, vol. 33, no. 2, pp. 1024–1034, 2001.

[18] B. Gutmann, I. Thon, A. Kimmig, M. Bruynooghe, and L. D. Raedt, "The magic of logical inference in probabilistic programming," *CoRR*, vol. abs/1107.5152, 2011.

[19] T. Sato, "A statistical learning method for logic programs with distribution semantics," in *ICLP*, 1995, pp. 715–729.

[20] D. Nitti, T. De Laet, and L. De Raedt, "A particle filter for hybrid relational domains," in *IROS*, 2013.

[21] L. S. Zettlemoyer, H. Pasula, and L. P. Kaelbling, "Learning planning rules in noisy stochastic worlds," in *AAAI*, 2005, pp. 911–918.

[22] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaebling, "Learning probabilistic relational planning rules," in *ICAPS*, 2004, pp. 73–82.